

Solutions for the Object-Relational Databases Design

Manole VELICANU, Iuliana BOTHA

Academy of Economic Studies, Bucharest

mvelicanu@yahoo.com, iuliana.botha@ie.ase.ro

The need for databases occurs in the moment when takes place an informatics system development. Moreover, databases are an important step in this process. For this reason, this paper deals with object-relational databases implementation as part of informatics systems development. The practical implementation is made on a decision support system (DSS) prototype, which can be applied in the uncertain and unpredictable environments, like the production and the prediction of the wind energy.

Keywords: *Object-Relational Databases, Informatics Systems Development, Unified Modeling Language (UML), Database Design, Database Implementation.*

1 Introduction

The IT&C fast development in recent decades is accompanied by significant changes in economic informatics. For this reason, the engagement of specialists is aimed at creating user-oriented information systems, which respond to requests promptly and accurately. Thus, over time, it was made the transition from computerized systems that address particular administrative problems, to intelligent systems which assist the decisions, based on knowledge modeling and processing.

Currently, organizations are required to store and process increasing quantities of data, requiring recourse to modern information technology, databases, data warehouses, Internet and, more so in these cases, the knowledge bases of intelligent systems.

Developing object-oriented model was due to inability of the relational model to

successfully deal with very large data volumes, of great complexity, encountered most often in new types of computer applications (multimedia, Internet, XML, spatial applications etc.). However, although OODBMS (Object-Oriented DBMS) appear to meet the needs for better software required by the new economy, markets for their use remains relatively low, the reason most often cited being the difficult query with a large consumption of computational resources.

The advantages and especially the limitations of both relational and object-oriented technology, demonstrate that businesses need the capabilities of both data models. As a result of these emerging technologies, it was developed the *object-relational data model*. This model provides an extension of the relational data model while also support the main concepts of the object-oriented one (Figure 1).

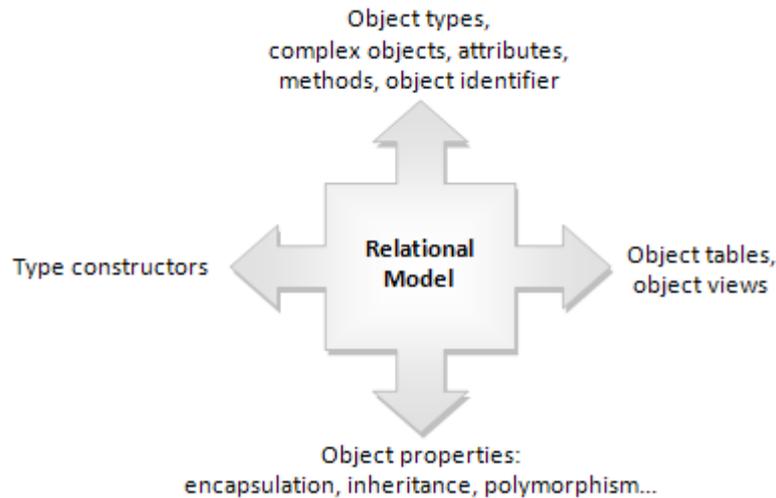


Figure 1 – Extensions of the relational model
(Source: Authors)

While in the 80s the relational databases were crowned as the most widely accepted databases, their limitations lead to a new hybrid database generation, according to the new data model specified above: object-relational databases.

The current technology has created the need for the use and storage of complex object types that were not supported by relational databases. Therefore, the new generation of database systems supports a unified relational and object-oriented data model. The idea is to import, as extensions of the relational model, the object-oriented concepts such as scalability, complex data types (large objects, multimedia data, spatial data, user defined object types etc.) and the fundamental object characteristics like encapsulation, inheritance and polymorphism [5].

2. Design of the object-relational databases

The *object-oriented methods* used for the design of the systems with object-relational databases are based on the concepts of object and classes of objects and allow the use of three different models for designing an object-relational database: the *static model* by which are modeled objects and the relations between them; the *dynamic model* by which are described interactions between objects; the *functional model* by

which are transformed data values using operations and processes.

The advantages of applying object-oriented methods in order to design the object-relational databases can be derived precisely from the facilities of unitary model behavior and data, and capture for the same object static, functional or dynamic characteristics, in terms of interactions with other objects. Thus can be modeled the business needs in interaction with objects.

In conclusion, we find that the benefits of the object-oriented methods in comparison with the structured one, recommend the object-oriented approach in the case of object-relational databases design.

Since object-oriented methodologies and methods have some limitations as well as many differences (in terms of symbols, notations or types of diagrams), it was needed a standard for modeling that can be widely applied in creating new systems or the maintenance of systems.

Thus, in 1997 was developed UML language (Unified Modeling Language), which brings standardization in the representation of symbols, notations, diagrams, and types of models which can be used for modeling a system with object-relational databases. In this way have been removed differences between object-oriented methodologies.

The latest version of UML is 2.3, which defines fifteen types of diagrams, divided into three categories. Thus, eight of them show static aspects, other three present general behaviors, and four represent different aspects of interaction between system elements.

3. UML extensions. Proposing an UML extension for object-relational databases design.

Modeling object-relational databases with object-oriented methods involves the usage of the basic concepts of object-oriented paradigm (classes, objects, encapsulation, aggregation, inheritance, polymorphism, abstraction) in implementing the static (structural) and dynamic model.

As stated in [4] one of the main goals of UML is to provide extensibility and specialization mechanisms by which basic concepts can be extended.

The standard set of UML notations and diagrams can be adapted using the following aspects:

- *Conditions of use.* Object-relational databases are designed especially for complex data storage (such as XML, multimedia, spatial) and must be optimized to achieve a variety of queries on them;
- *The data model used.* Object-relational databases implement the object-relational data model, a hybrid model that combines features of standard data models (relational and object-oriented);
- *Typical operations performed.* Using object-relational databases involves operations with multimedia data, spatial operations, and operations with structured or unstructured XML data.

To cover all modeling situations, the authors of UML [10], but also the OMG organization [12] give the possibility to extend syntax and semantics of UML

language through *stereotypes*, *comments* and *restrictions*.

Defining a collection of stereotypes, comments and restrictions, which extend an existing diagram type, in order to achieve a certain goal, is called a *profile*.

Together, the three mechanisms allow the creation of UML extensions adapted to a specific project. These mechanisms also allow adaptation of UML language to new information technologies, such as advanced object-oriented programming languages and their characteristics, distributed technologies, multidimensional analysis.

It is possible to add new items, to change the existing specifications and even to modify their semantics. Of course, it is important that these extensions to be done in a controlled manner, so that the UML objective remains unchanged, namely modeling information.

The study [3] presents a design methodology for object-relational databases. It defines new UML stereotypes, specific for object-relational databases and proposes a set of rules to transform UML schema in an object-relational one.

The papers [7] and [8] propose extending the set of standard UML stereotypes and restrictions by introducing a corresponding set for multidimensional modeling. For applications frequently used, like Web applications, [9] proposes an UML extension, and the studies [10] and [11] examine some extensions used to model relational databases.

Taking into account previous extensions for database design, [3] explains standard stereotypes of a relational database for each level: conceptual, logical and physical (Table 1).

Table 1 - Stereotypes for relational databases design

(Source: adapted from [3])

Database level	Database element	UML element	Stereotype
Conceptual level	Database	Component	<<Database>>
	Schema	Package	<<Schema>>
	Persistent class	Class	<<Persistent>>
	Multivalued attribute	Attribute	<<MA>>
	Calculated attribute	Attribute	<<DA>>
	Composite attribute	Attribute	<<CA>>
	Identifier	Attribute	<<ID>>
Logical level	Table	Class	<<Table>>
	Virtual table (view)	Class	<<View>>
	Column	Attribute	<<Column>>
	Primary key	Attribute	<<PK>>
	Foreign key	Attribute	<<FK>>
	Entity constraint (NOT NULL)	Attribute	<<NOT NULL>>
	Uniqueness constraint	Attribute	<<Unique>>
	Domain constraint (CHECK)	Constraint	<<Check>>
	Trigger	Constraint	<<Trigger>>
	Stored procedure	Class	<<Stored Procedure>>
Physical level	Tablespace	Component	<<Tablespace>>
	Index	Class	<<Index>>

As stated in [4], an UML extension must contain a short description, the list and description of stereotypes, the comments, the constraints, and also a set of formatting rules which are needed in order to specify if the data model is consistent.

For each stereotype are defined properties, semantics, comments and constraints. Also, each stereotype is identified by specific icons. *To define these stereotypes we propose the template* detailed in Table 2.

In order to use UML as modeling standard for object-relational databases, it should be extended to be applied to the object-

relational data model. It would be possible to represent special types such as collections (ARRAY), nested tables, references (REF).

We realized the UML extension using the CASE product IBM Rational Software Architect that offers the possibility to define custom profiles, which allows extending menus, stereotypes, comments and other restrictions. We applied the proposed extension to highlight the features of the object-relational model in the UML class diagram.

Table 2 - UML extension for object-relational database design
(Source: Authors)

<<UDT>>
- Name: User defined type
- Base class: Class
- Description: The extension defines an object class (an user defined object type)
- Image: 
- Constraints:
- It is used to define the data type of other elements
- Can encapsulate attributes and methods
- Values:
Unspecified
<<REF>>
- Name: Tip REF
- Base class: Attribute
- Description: The extension defines a relationship to another object type
- Image: 
- Constraints:
It is used to refer an object type
- Values:
The pointed object type
<<ROW>>
- Name: Tip ROW
- Base class: Attribute
- Description: The extension defines a composite attribute
- Image: 
- Constraints:
- It is used to refer a composite attribute type, with a given number of elements
- The elements of ROW type can have different data types
- The ROW type has no methods
- Values:
Name of each component and the corresponding data type
<<ARRAY>>
- Name: Tip ARRAY
- Base class: Attribute
- Description: The extension defines a collection data type
- Image: 
- Constraints:
- It is used in order to refer a collection data type, with a finite number of indexed elements
- The elements of ARRAY type can have any data type, except the ARRAY type
- Values:
Number of elements of the collection and the corresponding data types

<<OID>>	
-	Name: OID
-	Base class: Attribute
-	Description: The extension defines an unique identifier for each object type
-	Image: 
-	Constraints: Unspecified
-	Values: Unspecified

<<LOBAttribute>>	
-	Name: LOBAttribute
-	Base class: Attribute
-	Description: The extension defines a multimedia attribute (image, video, audio)
-	Image: 
-	Constraints: Unspecified
-	Values: Unspecified
	Derivation rule Type: LOB Multiplicity: 1

<<SpatialAttribute>>	
-	Name: SpatialAttribute
-	Base class: Attribute
-	Description: The extension defines a spatial (geographic) attribute of the object type
-	Image: 
-	Constraints: Unspecified
-	Values: Unspecified
	Derivation rule Type: Spatial Multiplicity: 1

<<UDTAttribute>>	
-	Name: UDTAttribute
-	Base class: Attribute
-	Description: The extension defines an attribute with the user defined type
-	Image: 
-	Constraints: Unspecified
-	Values: Unspecified
	Derivation rule Type: UDT Multiplicity: 1

<<Inheritance>>
- Name: Inheritance
- Base class: Association-Generalization
- Description: The extension defines a generalization relationship between object types
- Image: Unspecified
- Constraints: Unspecified
- Values: Unspecified

<<Validation rule>>
- Name: Validation rule
- Description: The extension defines the rules of membership of classes to object-relational data model
- Image: Unspecified
- Constraints: Unspecified
- Values: Unspecified

<<Supertype>>
- Name: Supertype
- Base class: Class
- Description: The extension defines a supertype from an inheritance hierarchy
- Image: 
- Constraints: Unspecified
- Values: Unspecified

<<Subtype>>
- Name: Subtype
- Base class: Class
- Description: The extension defines a subtype from an inheritance hierarchy
- Image: 
- Constraints: Unspecified
- Values: Unspecified

The above table shows a general pattern which can be used to indicate the specifications concerning the characteristics of object-relational model in the UML diagram of classes. Using the proposed extension, the classes diagram

will contain references and visual images of elements for the presented data model, such as object types, object identifiers, special attributes, and inheritance relationship.

4. Solution concerning the development flow of the applications with object-relational databases

Analyzing the *top-down* and *bottom-up* strategies for implementing the informatics systems, we conclude that, if using object-relational databases, it is appropriate to choose any of them based on the existing system characteristics.

A top-down approach is appropriate and desirable in a situation where there is not implemented an informatics system or if there are disparate applications which work as a result of local developments made over time, but their preservation is not vital.

If, however, what is already required to be kept in its current form and with current solutions, the approach should be bottom-up. This will try to integrate the existing systems into a new one which will be available throughout the organization.

After the identification and analysis of system requirements and setting its objectives we will decide whether to store some components of the existing system or replace them completely, by choosing an appropriate approach for development.

We consider that it is appropriate to achieve an unified and homogeneous system using top-down development strategy, considering that integrating existing databases it is very possible that they are not built using an object-relational data model and they use different DBMS.

Based on standard steps to be taken into consideration in order to develop database applications, we believe that the sequence of activities must be that shown in Figure 2.

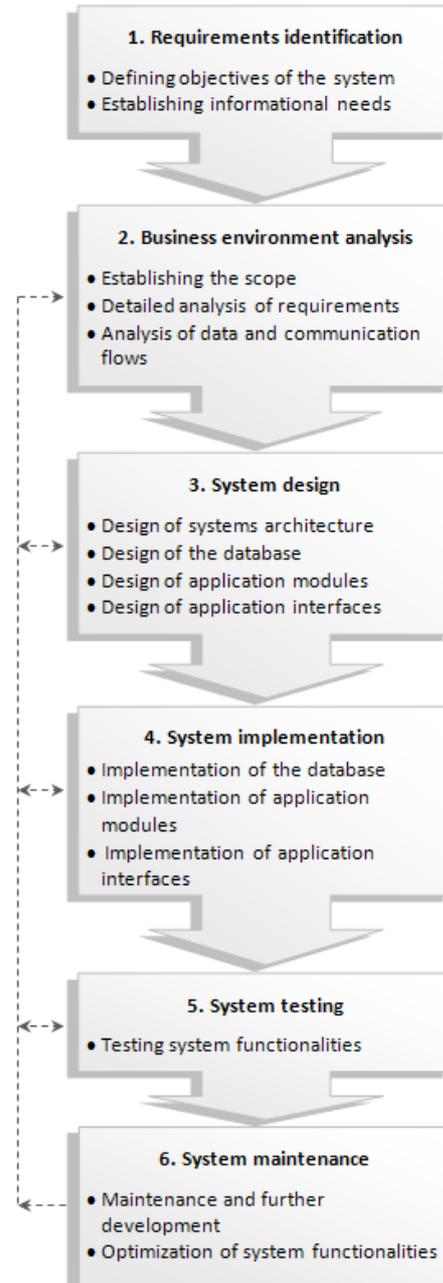


Figure 2 – Development flow of the applications with object-relational databases
(Source: Authors)

1) The first activity to be addressed is to *identify system requirements* that define the system objectives and future information needs for all its users. Must determine how to achieve the objectives and steps that are necessary to go, so there is a rigorous approach and the obtained system to work properly and have flexibility. Also, we need not to forget the study of law in the

fields of activity and the work rules and procedures established by the organization.

2) *Business environment analysis* involves the identification of the environment in which the business operates, the detailed analysis of operational requirements, the analysis of operating rules and procedures, data analysis and communication flow analysis within the organization and outside it.

3) The *system design* involves actually building the new system architecture, database design and corresponding applications design.

Database design is a very important step because it is responsible for ensuring consistency of data stored. A poor database design can lead to compromise data integrity, their redundancy and thus reduced performance.

Paper [13] identifies and describes three levels for designing object-relational databases, as we propose in Figure 2:

- *Conceptual level*, refers to the development of a model independently of any consideration regarding the appearance of data, as result of database analysis and modeling;
- *Logical level*, refers to the development of the object-relational data model, but independent of the chosen type of DBMS and other physical aspects of the model;
- *Physical level*, refers to the effective implementation of object-relational database, including these aspects related to ensure data security, managed through a specific DBMS.

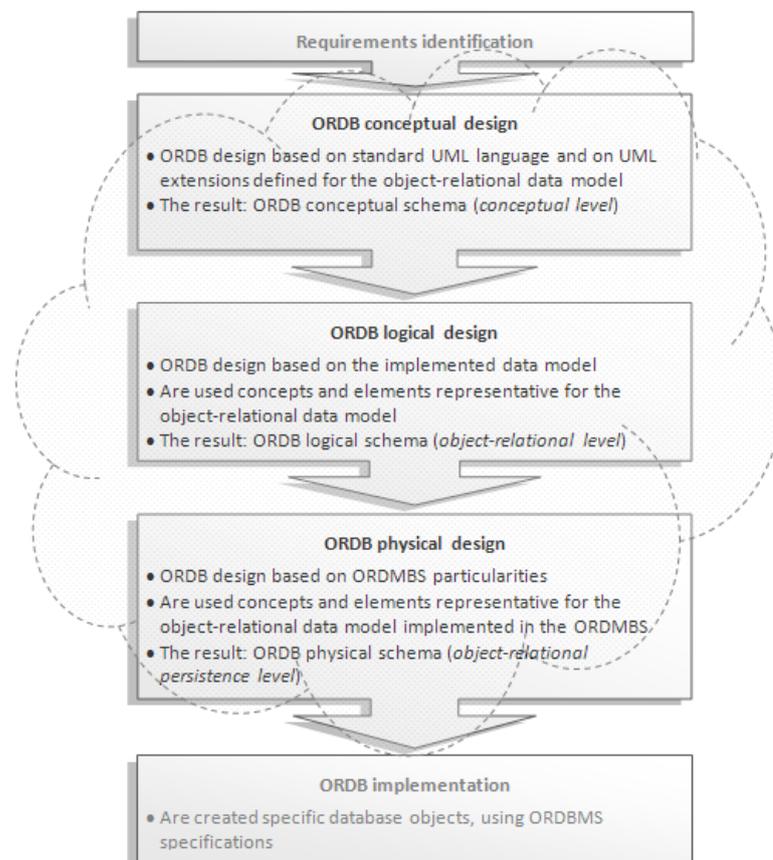


Figure 3 – Object-relational databases design
(Source: Authors)

Figure 3 shows the steps that are required to be made in order to design an object-relational database. In this respect, it is

necessary a process of mapping the model obtained from UML class diagrams to a standard object-relational model,

independent of product, and then the subsequent implementation in a DBMS. Mapping of the object model specific to UML language in an object-relational model that can be implemented in a database requires, according to [5], the implementation of identity, fields, classes, associations and relationships of generalization, aggregation and composition.

The identity problem is treated in [5], where two approaches are specified on the implementation of identity: based on the existence and based on value. The same paper also states aspects of data fields' implementation.

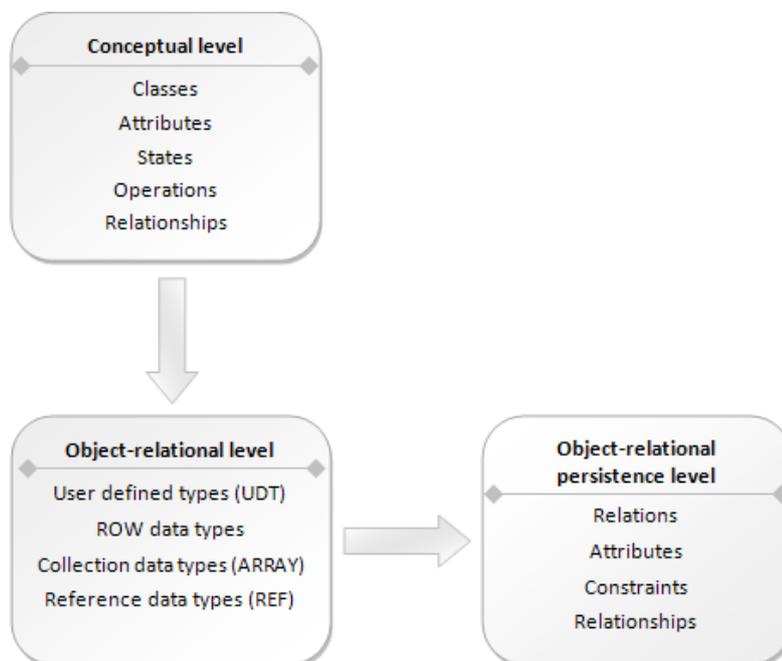


Figure 4 – Levels involved in object-relational mapping
(Source: adapted from [6])

Conceptual level of UML classes (conceptual schema of the database) meets the requirements of the system and focuses on classes, associations, attributes, states, operations. Can be identified different types of associations between classes, such as aggregation, composition, recursive association, hierarchy, class association.

Object-relational level (logical schema of the database) contains the elements proposed by the standard SQL:2003, such as: user-defined types, structured types, references, ROW data types and data

collections. Definitions made at this level do not allow persistent objects or data, until are created the tables that will store them.

According to [1], the transformation of an UML class diagram at object-relational is resumed to the mapping of logical structure of data and their behavior. This is not possible with traditional entity-association model (E-A). Unlike the relational model, the object types support encapsulation and the definition of methods can be explicitly associated with defining the types of objects.

Corresponding to the three stages of designing object-relational databases, [6] identifies three levels of mapping, outlined in Figure 3:

collections. Definitions made at this level do not allow persistent objects or data, until are created the tables that will store them.

Object-relational persistence level (physical schema of the database) is composed of tables, some of which are created from the previously defined object types. In addition to tables, this level also contains other specific relational elements, such as restrictions, data fields, relationships between tables.

In the practical experiment that we have realized, we used mapping to make the transition from the conceptual level, shown in the UML diagram of classes, at the object-relational one. Implementation is made in Oracle DBMS, by defining object types and data type constructors, corresponding to the object-relational data model. Later, we made the transition to the level of persistence, by creating tables of objects on the types defined in the previous level.

The paper [2] proposes to use UML class diagrams to design the conceptual schema instead of entity-association model (E-A), commonly used for relational databases.

Unlike E-A model, UML has the advantage that it allows the entire system design and makes easier the integration of different views on the system. Design phase is divided into two steps:

- Logical design, standard, independent of any product;
- Specific design, which takes into account a specific product (Oracle, Informix etc.).

The logical design is particularly important in designing object-relational databases, because each product implements a distinct object-relational model. Thus, we can identify an object-relational specification, independent of product, which can be easily determined using the standard SQL:2003.

In terms of graphical representation, based on the stereotypes proposed to extend UML, we can use a number of stereotypes specific for the object-relational model and also for the standard SQL:2003.

The specific design involves specifying the SQL schema in the chosen product. In terms of graphical representation, based on stereotypes proposed to extend UML, we can use a number of stereotypes specific for the object-relational model implemented in the selected product.

Next, from [3], we have indicated some rules to be followed for successful implementation of mapping between specified schemas (Table 3).

Table 3 – Rules used for mapping within schemas
(Source: adapted from [3])

UML	SQL:2003	Oracle 10g
Class	Structured type	Object type
Class extension	Typed table	Object table
Attribute	Attribute	Attribute
Multivalued attribute	ARRAY	VARRAY
Composite attribute	ROW / Attribute with structured type	Attribute with object type
Calculated attribute	Method / Trigger	Method / Trigger
Association		
1:1 association	REF-REF	REF-REF
1:n association	REF-ARRAY	REF-Nested table
n:n association	ARRAY-ARRAY	Nested table - Nested table
Aggregation	ARRAY	Nested table
Generalization	Data type	Child object type created with UNDER clause

In the practical application, we experienced the changes for the three levels of mapping presented. So we transformed the UML classes in Oracle object types and then their extensions in object tables. Associations between UML classes were identified as being of type one-to-many (1:n), which led to the inclusion of a REF type attribute in the type of object that participates in association with multiplicity n . Since some UML classes presented generalization relationships, we have achieved in Oracle the implementation of the concept of inheritance by defining subtypes (each object belongs to a sub-class, is also part of the super-class). We have made the inheritance definition by specifying the UNDER clause in the specifications of each subtype, indicating in this way the supertype.

The result of the design phase is represented by the object-relational database schema, used in the phase pre-implementation.

4) After the design, follows the *system implementation* phase, by which are specified the main components (using the components diagram) and their distribution on existing physical resources (modeled through the deployment diagram).

In this phase, is created the physical schema of the object-relational database, are written programs to define and manipulate objects in an ORDBMS that supports high-level programming languages, standard or proprietary, and also standardized declarative language SQL. Also, is tested the system and are made improvements in its operation.

Practical experimentation uses the design concepts defined above, making the changes necessary to transition from UML modeling to object-relational data model, and then to persistent objects in the database. We have implemented the concepts using the Oracle DBMS and the object extension included in the procedural language PL/SQL.

5) *Testing the system* enables its adjustment so we can decide its proper functioning regardless of the parameters involved. Thus, implementation and testing the system functionalities are phases that complete the system development.

6) However, during system operation will be made *maintenance and further development*, based on additional requests of the beneficiaries or based on changes in the organization activities. In case of failure, we can return to a previous activity and produce a new version of the system.

5. Conclusions

The paper deals with implementation issues concerning the object-relational databases, as part of informatics systems development.

Based on the literature and using the features of the standard modeling language UML, we have proposed an extension of it for modeling an object-relational database. The proposed template explains the standard stereotypes that are specific for the object-relational data model. For each stereotype are indicated properties, semantics, restrictions and comments.

Also, we have proposed a solution for the development flow of the applications with object-relational databases. Based on standard steps to be taken into account in the process of development of a database application, we consider that the sequence of activities must present specific features for each level, as outlined in the paper.

We considered the design as the most important activity by the fact that it is the only one that incorporates features of the object-relational data model. Thus, design is the activity that we detail the most in the paper, specifying a series of rules to be followed for a successful transition from standard design to the specific one.

Acknowledgment

This paper presents some results of the research project PN II, TE Program, Code 332: "Informatics Solutions for decision making support in the uncertain and

unpredictable environments in order to integrate them within a Grid network”, financed within the framework of People research program.

References

- [1] M.Wang, *Using UML for object-relational database systems development: A framework*, Issues in Information Systems, vol.9, no.2, 2008, ISSN 1529-7314
- [2] E.Marcos, B.Vela, J.M.Cavero, P.Cáceres, R.Juan, *Aggregation and Composition in Object – Relational Database Design*, 2001
- [3] E.Marcos, B.Vela, J.M.Cavero, *A Methodological Approach for Object-Relational Database Design using UML*, Software and Systems Modeling Journal, vol.2, no.1, 2003, pp.59–72, ISSN 1619-1374
- [4] A.Bâra, V.Diaconița, I.Lungu, M.Velicanu, *Improving Performance in Integrated DSS with Object Oriented Modeling*, WSEAS Transactions on Computers, no.4, vol.8, 2009, pp.599-609, ISSN: 1109-2750
- [5] C.Strîmbei, *Utilitatea metodologiei UML în proiectarea bazelor de date*, Informatica Economică Journal, no.2(26)/2003, pp.56-62, ISSN 1453-1305
- [6] M.F.Golobisky, A.Vecchietti, *Fundamentals for the Automation of Object-Relational Database Design*, IJCSI International Journal of Computer Science Issues, vol.8, no.2, 2011, ISSN 1694-0814
- [7] S.Luján-Mora, J.Trujillo, I.Y.Song, *Extending UML for Multidimensional Modeling*, vol. Proceedings of the 5th International Conference on The Unified Modeling Language, 2002, pp.290-304, ISBN 3-540-44254-5
- [8] S.Luján-Mora, J.Trujillo, I.Y.Song, *A UML profile for multidimensional modeling in data warehouses*, Journal Data & Knowledge Engineering - Special issue, vol.59, no.3, 2006, pp.725-769, ISSN 0169-023X
- [9] J.Conallen, *Building Web Application with UML*, Addison-Wesley, 2000, ISBN 0-201-61577-0
- [10] G.Booch, J.Rumbaugh, I.Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1998, ISBN 0-201-57168-4
- [11] E.J.Naiburg, R.A.Maksimchuk, *UML for Database Design*, Addison-Wesley, 2001, ISBN 0-201-72163-5
- [12] OMG, *Unified Modeling Language: Infrastructure and Superstructure*, August 2007, <http://www.uml.org/>
- [13] T.Connolly, C.Begg, *Database Systems. A Practical Approach to Design, Implementation and Management*, 4th edition, Addison Wesley, 2004, ISBN 0-321-210-255



Manole VELICANU is a Professor at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. He has graduated the Faculty of Economic Cybernetics in 1976, holds a PhD diploma in Economics from 1994 and starting with 2002 he is a PhD coordinator in the field of Economic Informatics. He is the author of 18 books in the domain of economic informatics, 64 published articles (among which 2 articles ISI indexed), 55 scientific papers published in conferences proceedings (among which 5 papers ISI indexed and 7 included in international databases) and 36 scientific papers presented at conferences, but unpublished. He participated (as director or as team member) in more than 40 research projects that have been financed from national research programs. He is a member of INFOREC professional association, a CNCSIS expert evaluator and a MCT expert evaluator for the program *Cercetare de Excelenta - CEEEX* (from 2006). From 2005 he is co-manager of the master program *Databases for Business*

Support. His fields of interest include: Databases, Design of Economic Information Systems, Database Management Systems, Artificial Intelligence, Programming languages.



Iuliana BOTHA is an Assistant Lecturer at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. She has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2006 and the Databases for Business Support master program organized by the Academy of Economic Studies of Bucharest in 2008. Currently, she is a PhD student in the field of Economic Informatics at the Academy of

Economic Studies. She is co-author of 6 books, 13 published articles (3 articles ISI indexed and the other 10 included in international databases), 16 scientific papers published in conferences proceedings (among which 6 paper ISI indexed). She participated as team member in 4 research projects that have been financed from national research programs. From 2007 she is the scientific secretary of the master program *Databases for Business Support* and she is also a member of INFOREC professional association. Her scientific fields of interest include: Databases, Database Management Systems, Design of Economic Information Systems, Business Intelligence, e-Learning Technologies.